How do I automate in Selenium?

Selenium Automation

Selenium is still the most popular and well-developed web automation testing framework. It is positioned to lead the industry in all aspects compared to other trending frameworks like Cypress, Puppeteer, Playwright, and others. It is one of the most excellent automation frameworks with constantly expanding capabilities. Furthermore, Selenium allows you to utilize other programming languages for Selenium automation testing, such as C#, Ruby, Perl, Java, Python, and others, and different operating systems and web browsers.

Selenium is a set of tools that lets you write automated tests for web applications in various programming languages. The Selenium project is maintained on GitHub by SeleniumHQ and includes various open-source Selenium projects.

The Selenium IDE, Selenium WebDriver, and Selenium Grid are all part of the Selenium suite, each having its approach to browser automation.

Automation testing demands a thorough understanding of various automation technologies and frameworks. Selenium is the most popular of these tools due to its ease of use and useful features. Selenium is a tool for automating website testing and monitoring their performance.

Prerequisites for Selenium Automation

Skills in coding

The majority of testers are unable to code. However, even if you're one of the few that does, the time it takes to set up and manage your test cases will rapidly outstrip the time to run the tests manually.

The rest of the pipeline's integration

You'll need to make sure that your code speaks with the rest of the release pipeline and set up and update the script. As a result, you'll need to understand how your Git repository handles code versioning as a tester.

Creating suitable test environments

Setting up your test environment is a difficult task in and of itself. Setting up parallel test environments and cross-browser testing is achievable using Selenium, but it can take months even for a talented coder.

Maintenance

You'll have to maintain your scripts in addition to writing them. You might be able to detect and correct any problems in your script if you're good with code and have a very skilled programmer with a lot of free time.

Automation steps with Selenium

There are a few primary stages to writing a test case for the login functionality that must be followed before starting automation testing:

1. Make a Selenium WebDriver application

Specify the system properties to the path of the required browser's driver to run the website in that browser. In this example, Selenium Webdriver will be used to automate logins using ChromeDriver.

The following is the syntax for the same:

```
Webdriver driver = new ChromeDriver();
System.setProperty("webdriver.chrome.driver", "Path of the chrome driver");
```

2. Configure your browser if necessary

When the test case is run, the web page is usually in a reduced format. Increase the size of your browser to get a better view of the test cases you've run. To do the same, type the following command:

```
driver.manage.window.maximize();
```

3. Go to the necessary web page

Open the desired URL in the browser. For example, to open the URL in the preferred instantiated browser, type the following command:

```
driver.get("https://www.browserstack.com/users/sign_in");
```

4. Find the appropriate web element

Locators are an essential component of any Selenium script since they indicate the elements with which the test script will interact to duplicate user actions.

Locate it via the ID locator in Selenium WebDriver:

```
driver.findElement(By.id("user_email_login"));
```

Since this returns a webelement, store it in webelement variable with:

```
WebElement username=driver.findElement(By.id("user_email_login"));
```

Repeat the same steps for the password field.

5. Perform Action on the Located Web Element

Testers must then perform the desired action after locating the element. The action in this example is to type text into the email and password fields before clicking the login button.

The script accomplishes this by employing the sendKeys and click methods, as demonstrated below:

```
username.sendKeys("abc@gmail.com");
password.sendKeys("your_password");
login.click();
```

6. Verify & Validate the Action

Assertion can be used to verify the findings. The use of assertions in comparing predicted and actual results is critical. The test case passes if it matches. If not, the test case will fail.

The final assertion will look like

```
Assert.assertEquals(actualUrl, expectedUrl);
```

Conclusion

Selenium is a free, open-source project that offers a variety of software tools for performing automated testing. Its key characteristic is that it can be used to run tests across many operating systems and web browsers with ease. Selenium is widely used by testers and QA managers to automate the most repetitive and dull testing activities, and for a good reason. It is a popular tool that can improve the accuracy and dependability of your tests if used correctly. In addition, Selenium is compatible with various browsers, including Internet Explorer, Chrome, Firefox, Opera, and Safari. Running tests and testing them across multiple browsers simultaneously becomes useful.